AMENDMENTS TO THE SPECIFICATION

Please replace paragraph [0009] with the following amended paragraph:

[0009] At least one problem with this manner of testing a program is that it can be fairly difficult for a tester to think of every possible way in which a program can be tested. For example, a calculator program could have an error that only occurs with a limited subset of input values. If the tester is manually testing individual numbers, it would be inefficient, [[and/ or]] and/or impossible for the tester to think of every number and every available mathematical operation in order to flesh out any programmatic errors. In some cases, such as with "test automation", the software developer may write a secondary testing program that the tester could use to test for many of the possible inputs in a relatively short time. While this can provide some benefits, it can be a burden to write program-specific tests for each newly written program.

Please replace paragraph [0021] with the following amended paragraph:

[0044] Figure 4 illustrates a method for testing an application program with respect to results obtained for one or more user interfaces in accordance with at least one embodiment of the present invention; and

Please replace paragraph [0028] with the following amended paragraph:

[0028] Generally, an interface, such as a GUI 130, will comprise executable program code that allows a user to request a relevant computerized system to perform a task. In typical operation, a user requests a task through the interface by some physical act such as by entering keystrokes through a keyboard, speaking a command, or by selecting an actionable item with a mouse

device. In turn, each such user interface will execute the user's valid requests by executing any variety of application program interfaces ("API"), which [[are]] can often be interface-specific. These sets of APIs can include system or interface-specific APIs, as well as APIs that are specific for accessing an application program, such as accessing a calculator program remotely over a telephone interface.

Please replace paragraph [0029] with the following amended paragraph:

[0029] For example, telephone 120 may implement a set of APIs 122 such as API 110 and API 112. Graphical user interface 130 may alternatively implement a set of APIs 132 such as API 110, API 134, and API 136. Similarly, personal digital assistant 140 may implement a set of APIs 142 such as API 110, API [[144]] 134, API [[146]] 145, and command line interface 150 can implement a set of APIs 152 such as API 110, API 154, and API 156. API 110 may be specific for accessing a calculator or date program through a wireless protocol. As well, API 134 may be specific for displaying the output of the calculator or date program within a graphical interface. However configured, the user interface API's implement the user's request directly to an application program, such as by making a function call to an application program that is accessible through the relevant computer's operating system.

Please replace paragraph [0033] with the following amended paragraph:

[0033] By way of further explanation, the number and type of isomorphisms for a given value can be unique from one interface to the next. In particular, one interface may be able to receive or understand five ways of representing a given value, another interface might be able to receive or understand twelve ways of representing the given value, and yet another interface might be

able to receive or understand only one or two ways of representing the given value. With reference to the previously described date example, a calendar interface that receives text as the primary user input may be configured to understand as many as 5 textual ways of representing the date December 19, including both numeral ("12") and spelled-out ("December") representations of the month portion of the date. By contrast, a voice-based interface that implements the same API may be configured to receive or understand only the commonly spoken versions of the date value, such as "December nineteen", "December nineteenth", "nineteenth of December", and so forth, as opposed to "twelve nineteen". As such, in at least one implementation of the present invention, the test program can automatically identify isomorphisms 207 of a given value 205 that are interface-specific.[[.]]

Please replace paragraph [0036] with the following amended paragraph:

[0036] Another aspect of the present invention is that a test program that is written for one API, for example test program 210 written for API 110, does not need to be re-written for any additional given API, such as API 134, which, in accordance with Figure 1, is common only to user interfaces 130 and 140. The test program 210 can be converted in a conversion module 230 that converts the test program so that it can be run with the different API. In one embodiment, the conversion module 230 takes the source code of the test program 210 and recompiles the source code of the test program 210 such that the test program 210 becomes test program 212, which is built around the new API 134, and/or is configured to test new Application Program 202. Alternatively, conversion module 230 can convert an API call corresponding to a first API for compatibility with a second API at runtime.[[.]] Similar embodiments are possible for test programs that implement scripting languages rather than compiled source code. In any case, a

tester can implement test programs, e.g., 210, 212, for multiple APIs and multiple Application Programs without rewriting the test program 210 from scratch.

Please replace paragraph [0045] with the following amended paragraph:

[0045] Although step 440 can comprise any number or combination of corresponding acts for accomplishing the step 440 depicted in Figure 4, step 440 comprises act 420 of receiving a plurality of results from the application program. Act [[440]] 420 includes receiving a plurality of results from the application program, wherein each result in the plurality corresponds to an identified one of the plurality of interfaces. For example, a tester may run a developed test program 210 that has been configured around the identified, common API 110 multiple times, or may further implement the conversion module 230 so that the test program 210 can be configured to mimic other interfaces or APIs (e.g., 134, etc.) and other application programs (e.g., 202) more closely. While, in some embodiments, this can require additional testing, rather than running few or one tests for a given API, this should nevertheless not require rewriting of the test program. For example, a test program 210 can be recompiled automatically into test program 212 for a different interface before running the test. Hence, time and effort is still significantly reduced since the test program 210 does not necessarily need to be rewritten by the tester.